UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/713,872	11/14/2003	Matthias Eberhard Sohn	11884/405801	1867

26646 7590 08/08/2007

KENYON & KENYON LLP
ONE BROADWAY
NEW YORK, NY 10004

EXAMINER	
VU, TUAN A	
ART UNIT	**PAPER NUMBER**
2193	

MAIL DATE	DELIVERY MODE
08/08/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/713,872	SOHN ET AL.	
	Examiner	Art Unit	
	Tuan A. Vu	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *29 June 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *2,4,5,10,12,15 and 17-24* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *2,4,5,10,12,15 and 17-24* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *updated copy of 3/05/07*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 6/29/07.

As indicated in Applicant's response, claims 2, 4-5, 10, 12, 15 have been amended, claims 1, 3, 6-9, 11, 13-14, 16 canceled, and claims 17-24 added. Claims 2, 4-5, 10, 12, 15, 17-24 are pending in the office action.

Specification

2. The listing of the Drawings, pg. 5, appears to have omitted Figures 3b-c, whereas the Specifications (pg. 8-9) and the Drawings actually mention about or include these Figures.

Appropriate correction is required.

Claim Objections

3. Claim 18 is objected to because of the following informalities: there appears to be a typo mistake in the double ";" at the end of 'and provide functionality for the application'(line 9). Appropriate correction is required.

Claim Rejections - 35 USC § 112

4. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

5. Claim 2, 4-5, 10, 12, 15, 17-24 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

6. Claims 17-24 recite the limitation "application framework in the second layer" (cl. 17, li. 7; cl. 18, li. 7; cl. 19, li. 12; cl. 20, li. 13; cl. 21, li. 7). There is insufficient antecedent basis for

this limitation in the claim. This second layer limitation will be treated as 'application framework'.

7. Claims 17 recites 'responsive to a change in an application framework, dynamically modeling an application object repository framework in *a first layer* ... using ... constructs ... defined by a **repository framework model** in ***a second layer***; generating application framework metadata representing the application framework ... occupying *the first layer* ... using the ... constructs defined by the **repository framework model** in ***the first layer***'; and there is a large inconsistency as to what entity pertains to a first layer as opposed to being a format in a second layer: e.g. is repository framework model second or first layer? whether application object repository framework is first layer, or application framework metadata is first layer, or repository framework model being first layer. The Specifications do not provide a succinct and clear description as to how these layers are defined for the above inconsistent language to teach one of ordinary skill in the art to make use of the invention as claimed.

Claims 2, 4-5, 10, 12, 15, 22-24 are also rejected for not remedying to the above.

8. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

9. Claims 2, 4-5, 10, 12, 15, 17-24 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention.

Claim 17 recites 'responsive to a change in an application framework, dynamically modeling an **application object repository framework** in *a first layer* ... ; generating **application framework metadata** representing the application framework ... occupying *the first layer* ... as meta-model ... using the ... constructs defined by the **repository framework model**...'. There is no mention of *application object repository framework* (AORF) being modeled dynamically in response to a change in application framework in the Specifications **prior to** generating an application framework metadata (AFM) as metamodel such that this will be occupying the first layer; and one of ordinary skill in the art would be unable to construe the likes of: whether a step of modeling (in response to a change) exists before the generating of application framework metadata; whether this AORF modeling step encompasses the generation of AFM metamodel; whether the AORF and the AFM model are meant to be distinct OR represent distinct stages. What appears to be insufficient an enablement by the Disclosure is due in part because there is no consistency in terminology for conveying what characterizes the stages and the entities involved in the course deriving changes to the application framework from the recited steps; notably when the inventor seems not able to provide a clear and succinct description for each of said action stage that would clearly map each terminology being claimed in a steady manner. The 'Summary of the Invention' mentions about a AORF that can be modeled (Specifications, pg. 4 top line); but does not provide specifics as to whether it is done prior to a AFM model generation, or responsive to something, e.g. a input or a form of specification as basis for the modeling. For one attempting to make use of the invention, it is unclear as to whether the inventor provides sufficient features to support the modeling of a AORF in response to change (being identified/detected) and then generating of AFM model to

occupy the first layer (notwithstanding the 35 USC § 112, 2nd paragraph rejection), absent of a deliberate mention as to this *application object repository framework* entity (or AORF) and modeling thereof in response to said *change*. In view of the above lack of enabling disclosure, the modeling of AORF will be treated as little as a modeling environment where first layer components are defined with a second layer semantics the language of which is defined by a common language in a third layer; along with the rest of the claim interpreted as obtaining of AFM and transformation thereof into some other repository intermediate form; that is, the AORF modeling limitation (i.e. dynamically; responsive to … change) not treated with barely any patentable weight.

Claim 18 also recites 'responsive to a change in an application framework, dynamically modeling an application object repository framework … in a first layer; receiving a UML representation of the application framework metadata … occupying the first layer … ; upon said receiving, transforming the application framework metadata into XML repository framework metadata …'. One would not be taught how dynamically (dynamic with respect to what ?) modeling of a AORF responsive to a change is conveyed **prior to** the receiving and transforming of the AFM, when there is no definite teaching in the Specifications mentioning about co-existence of both AORF and AFM such that the former is modeled prior to the obtaining of the latter, leading to creating of repository framework metadata. It is evident that the once-mentioned AORF is not mentioned anywhere else in the claim; and it becomes apparent as to whether the AORF is not misnomer to merely represent a environment encompassing the substep of obtaining AFM and transforming this into XML, absent any clear definition of AORF as set forth above (refer to claim 17) any where in the Specifications. One would not be taught about

the existence of this AORF modeling and the subsequent AFM generation and would be hard pressed to make use of the invention. The modeling of AORF will be treated as a modeling framework including obtaining of AFM and transformation thereof; that is, the **dynamically modeling** of AORF treated with minimal patentable weight.

Claim 19 recites '... responsive to a change ... dynamically model an application object repository framework in a first layer ... transform the received application framework metadata into XML...' and is also rejected for the reasons set forth for claim 18.

Claims 20 and 21 recite '... responsive to a change ... dynamically modeling ... transforming ... application framework meta-data ...'; hence are also rejected for not providing sufficient teaching from the Specifications to enable one of ordinary skill in the art to make use of the invention.

Claims 2, 4-5, 10, 12, 15, 22-24 for failing to remedy to the above are also rejected; and the modeling limitation in regard to the AORF will be given weight only to the extend of how the transforming or obtaining of AFM as claimed is being interpreted.

Claim Rejections - 35 USC § 103

10. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

11. Claims 2, 4-5, 10, 12, 15, and 17-24 rejected under 35 U.S.C. 103(a) as being unpatentable over Iyengar, USPN: 6,874,146 (hereinafter Iyengar).

As per claim 17, Iyengar discloses a method for generating a software development repository to reflect extensions in an application framework that supports an application, the method comprising:

in a multi-layer modeling architecture, responsive to the application framework extensions, dynamically modeling an application object repository framework (e.g. Fig. 3 – Note: runtime framework to address request coming as first layer application object being modeled via ORB –see *request* -Fig. 1 – reads on first layer constructs being modeled with runtime framework based on a request, i.e. the application request being in a first layer ORB metamodel) in a first layer using repository constructs and semantics defined by a repository framework model in a second layer (e.g. *metamodel UML 21* Repository – Fig. 1 – Note: language describing UML reads on second layer semantics- see Fig. 4), wherein the repository framework model is defined by a common modeling language in a third layer(e.g. MOF 20, Fig. 3) that also models the application framework in the second layer, wherein the application framework supports the application by providing application constructs and semantics to structure and provide functionality for the application, and wherein the application framework extensions provide additional functionality to the application framework (Fig. 3; col. 9, line 40 to col. 10, line 19; *extending this capability* - SUMMARY col. 3);

generating application framework metadata representing the application framework and (*metamodel* UML 21 - Fig. 2; UML- Fig. 4) occupying the first layer (e.g. Fig. 3) in the multi-layer modeling architecture as meta-model data using the repository constructs defined by the repository framework model (Fig. 3) in the first layer;

upon said generating, validating the generated meta-model data with respect to the repository framework model constructs (e.g. col. 9, line 40 to col. 10, line 19; *mapping rules* - col. 10, line 41 to col. 42);

upon said validating, transforming the meta-model data into repository framework metadata, the repository framework metadata representing an intermediate representation (e.g. *XML ...produced* - col. 9, lines 24-32) of an application object repository and occupying the first layer of the modeling architecture;

transforming the repository framework metadata into application object repository source files the source files including a runtime source file and generating an application object repository from the application object repository source files (*extensible object-oriented database application* – col. 7, lines 61-65; see col. 8: *incorporated by reference* col. 8 - USPN: 5,644,764, Johnson et al: e.g. *service 22a, 22b, 22c, 22d*– Fig. 1; col. 3, line 3 to col. 4, line 24)

But Iyengar does not explicitly disclose application object repository source file using a predefined transformation template, and a database schema script. However, Iyengar's suggested use of XSL (col. 3, lines 45-52) serving as superior constructing tool to build application from a XML schema entails a script being based upon a XSL template. In view of the teaching from USPN: 5,644,764 regarding services to persist OO data in what appears to be an extensible database as by Iyengar (see col. 7, lines 61-65), one of ordinary skill in the art would utilize this superior use XSL template approach to make use of the XML metadata in order to implement by means of script the extension and development services contemplated by Iyengar in view of the teachings by Johnson (incorporated by reference) to provide persistence of objects and extensibility to a database using Object-Oriented tracking by type, attribute, methods

and property as by Johnson, such that the OO constructs can be represented as schema of tagged objects as contemplated by Iyengar's XML format (see col. 6, line 52 to col. 7, line 50)

Iyengar does not specifically teach generating an application object repository schema from the database schema script, the application object repository schema defining a relational database structure for storing application metadata representing the application framework extensions. However, based on the XSL approach and a scripting to provide services supporting a OO extension of database and reusing of OO components via such script language as addressed above, with a construct of a XML representing a DB form of schema (see col. 6, line 52 to col. 7, line 50), the SQL script for supporting relational database structure by Johnson or Iyengar's DB extension would have been obvious for the same reasons as set forth above.

Iyengar does not explicitly disclose compiling the runtime source file to generate an executable component, the executable component providing at least one database service for object-oriented interaction with the stored application metadata in the application object repository.

However, based on the rationale to provide script and OO constructs to support extension of database as taught above (Iyengar: col. 7, line 60 to col. 8, line 45; *developers of ...repositories* , non-Corba ...applications,... development of Corba-based software, database interoperability - col. 9, lines 21-67; - *incorporated by reference*, col. 8; i.e. USPN: 6,018,627 -- hereinafter Iyengar2: *build components, assemble, deploy* –Fig. 2B), one of ordinary skill in the art would be motivated to implement runtime source file supporting this object repository schema and relational structure storage compliant with the services as contemplated by Johnson and extension by Iyengar (see above) such that these source files would be compiled into

executable or deployable form as by Iyengar2 because the main purpose of having metadata repository by Iyengar is to support database OO application, interoperability between database (see col. 9, lines 21-67), to extend its functionality (col. 7, lines 61-65) via persistence (see Johnson) and in providing OO metadata to support applications as set forth above, the concept of providing of runtime executable (see Iyengar2) would have been obvious because this necessary means would make use of runtime components assembled via the XML script construct or schema deploying the services as set forth above (see Iyengar: col. 7, line 60 to col. 8, line 45; USPN: 5,644,764: Fig. 1-7; USPN: 6,018,627: Figs 1-3).

As per claim 2, Iyengar discloses wherein the repository framework metadata is XML ("Extensible Markup Language"): see Fig. 2-4.

As per claims 4-5, Iyengar does not explicitly disclose wherein the at least one service includes versioning, object-oriented access, persistence and change management. But based on the extensibility of Iyengar's database (see col 7, lines 61-65) and the services by Johnson (incorporated by reference USPN: 5,644,764: Fig. 1-7), the DB versioning, access, persistence and change management services would have been obvious services according to the rationale as set forth in claim 17.

Nor does Iyengar explicitly disclose wherein said transforming the meta-model data application into repository framework metadata is achieved using XSL ("Extensible Style Language"). However, the use of scripting to make use of the extensible XML metadata format has been suggested in Iyengar, and rendered obvious in view of the template-based scripting limitation as set forth in claim 17.

As per claim 18, Iyengar discloses a method for generating a software development repository to reflect changes in an application framework that supports an application, the method comprising:

in a multi-layer modeling architecture, responsive to a change in an application framework, dynamically modeling an application object repository framework (e.g. Fig. 3 – Note: runtime framework to address request coming as first layer application object being modeled via ORB –see *request* -Fig. 1 – reads on first layer constructs being modeled with runtime framework based on a request, i.e. the application request being in a first layer ORB metamodel) in a first layer using repository constructs and semantics defined by a repository framework model in a second layer (e.g. metamodel UML 21 Repository – Fig. 1 – Note: language describing UML reads on second layer semantics- see Fig. 4), wherein the repository framework model is defined by a common modeling language in a third layer (e.g. MOF 20, Fig. 3) that also models the application framework in the second layer, wherein the application framework supports the application by providing application constructs and semantics to structure and provide functionality for the application (e.g. Fig. 3; col. 9, line 40 to col. 10, line 19);

receiving a UML representation (e.g. *metamodel* UML 21 - Fig. 2; UML- Fig. 4) of application framework metadata representing the application framework and occupying the first layer (e.g. Fig. 3) in the multi-layer modeling architecture, the application framework metadata specified utilizing predefined UML constructs and repository constructs defined by the repository framework model in the second layer (e.g. col. 9, line 40 to col. 10, line 19);

upon said receiving, transforming the application framework metadata into XML repository framework metadata representing an intermediate representation of the application object repository and occupying a first layer of the modeling architecture, the repository framework metadata being a function of the repository framework model and the common modeling language (e.g. *XML ...produced* - col. 9, lines 24-32; *mapping rules* - col. 10, line 41 to col. 42);

transforming the XML metadata into application object repository source files including runtime source file and generating an application object repository from the application object repository source files (e.g. *extensible object-oriented database application* – col. 7, lines 61-65; see col. 8: *incorporated by reference* - USPN: 5,644,764, Johnson et al: e.g. *service 22a, 22b, 22c, 22d*– Fig. 1; col. 3, line 3 to col. 4, line 24)

Iyengar does not explicitly disclose transforming the XML repository framework metadata into application object repository source files using a predefined XSL transformation template, the source file including a database schema script; however, Iyengar's suggested use of XSL (col. 3, lines 45-52) serving as superior constructing tool to build application from a XML schema entails a script being based upon a XSL template. In view of the teaching from USPN: 5,644,764 regarding services to persist OO data in what appears to be an extensible database as by Iyengar (see col. 7, lines 61-65), one of ordinary skill in the art would utilize this superior use XSL template approach to make use of the XML metadata in order to implement by means of script the extension and development services contemplated by Iyengar in view of the teachings by Johnson (incorporated by reference) to provide persistence of objects and extensibility to a database using Object-Oriented tracking by type, attribute, methods and property as by Johnson,

such that the OO constructs can be represented as schema of tagged objects as contemplated by Iyengar's XML format (see col. 6, line 52 to col. 7, line 50)

Iyengar does not specifically teach generating the above application object repository from the above application object repository source files, **so to further generating an application object repository schema** from the database schema script using OSQL ("object-oriented SQL"), the application object repository schema defining a relational database structure for storing application metadata representing application development objects and the relations between the application development objects compliant with the application framework changes.

However, based on the XSL approach and a scripting to provide services supporting a OO extension of database and reusing of OO components via such script language as addressed above, with a construct of a XML representing a DB form of schema (see col. 6, line 52 to col. 7, line 50), the SQL script for supporting relational database structure by Johnson or Iyengar's DB extension would have been obvious for the same reasons as set forth above.

Iyengar does not explicitly disclose compiling the runtime source file to generate an executable component, the executable component providing at least one database service for object-oriented interaction with the stored application metadata in the application object repository; but based on the rationale to provide script and OO constructs to support extension of database as taught above (Iyengar: col. 7, line 60 to col. 8, line 45; *developers of ...repositories* , non-Corba ...applications,... development of Corba-based software, database interoperability - col. 9, lines 21-67; - *incorporated by reference*, col. 8; i.e. USPN: 6,018,627 --hereinafter Iyengar2: *build components, assemble, deploy* –Fig. 2B), one of ordinary skill in the art would be motivated to implement runtime source file supporting this object repository schema and

relational structure storage compliant with the services as contemplated by Johnson and extension by Iyengar (see above) such that these source files would be compiled into executable or deployable form as by Iyengar2 because the main purpose of having metadata repository by Iyengar is to support database OO application, interoperability between database (see col. 9, lines 21-67), to extend its functionality (col. 7, lines 61-65) via persistence (see Johnson) and in providing OO metadata to support applications as set forth above, the concept of providing of runtime executable (see Iyengar2) would have been obvious because this necessary means would make use of runtime components assembled via the XML script construct or schema deploying the services as set forth above (see Iyengar: col. 7, line 60 to col. 8, line 45; USPN: 5,644,764: Fig. 1-7; USPN: 6,018,627: Figs 1-3)

As per claim 10, Iyengar does not explicitly disclose wherein the at least one service includes versioning, object-oriented access, persistence and change management. But based on the extensibility of Iyengar's database (see col 7, lines 61-65) and the services by Johnson (incorporated by reference- col. 8- USPN: 5,644,764: Fig. 1-7), the DB versioning, access, persistence and change management services would have been obvious services according to the rationale as set forth in claim 18.

As per claim 19, Iyengar discloses a system for generating an object repository to reflect changes in an application framework, the system comprising:

an interface to receive a visual representation of application framework metadata representing an application framework (e.g. Fig. 3); a processor; and

a memory, coupled to the processor, storing instructions adapted to be executed by the processor to perform:

(...responsive to a change in an application framework), dynamically model an application object repository framework in a first layer (... semantics defined by a repository framework model in a second layer, wherein the repository framework model is defined by a common modeling language in a third layer ... provide functionality for the application);

transform the received application framework metadata into XML ... an intermediate representation (... of the application object repository and occupying a first layer ... being a function of the repository framework model and the common modeling language;

transform the XML repository framework metadata into application object repository source files ... generate an application object repository from the application object repository source files;

generating an application object repository schema (from the database schema script using ... application object repository schema defining a relational database structure ... the application framework changes); and

compiling the runtime source file (to generate an executable component, the executable component providing at least one database service ... the application object repository;

all of which step limitations having addressed in claim 18.

As per claim 12, Iyengar discloses wherein the visual representation of the application framework metadata utilizes at least a subset of UML ("Unified Modeling Language"): see *metamodel* UML 21 - Fig. 2; UML- Fig. 4.

As per claim 20, Iyengar discloses system to generate an object repository to providing generic migration of previously stored data in a software development repository to reflect changes in an application framework, the system comprising: an interface to receive application

framework metadata representing an application framework; a processor; and

a memory, coupled to the processor, storing instructions adapted to be executed by the processor (refer to claim 19) to perform the same step limitations as set forth in claim 19, including the obvious teachings as set forth therein.

As per claim 15, Iyengar discloses further comprising a database storing versions of the application object repository to provide migration of data stored in the application object repository (e.g. SUMMARY: *data interchange ... among repositories* -col. 3, li. 61-67; col. 9, lines 55-56; see incorporated by reference, col. 8: USPN: 5,644,764: Fig. 1-7 – Note: versioning and interoperability between repositories using extensible metadata support reads on migration of data via interchange between DB format).

As per claim 21, Iyengar discloses a method for providing generic migration of previously stored data in a software development repository to reflect changes in an application framework, the method comprising the same step limitations as set forth in claim 17, including the obvious teachings as set forth therein.

As per claims 22-24, Iyengar discloses wherein the application object repository source files are C++ files (incorporated by reference, col. 8: USPN: 6,018,627 – Iyengar2: see col. 9, lines 30-64).

Response to Arguments

12. Applicant's arguments with respect to the submission filed 6/29/07 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

13. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (571)272-3756.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Tuan A Vu
Patent Examiner,
Art Unit 2193
August 05, 2007